Continual Learning with Class-Incremental MNIST

Saurabh Shetty[1], **Siddharth Mittal**[1],

[1]Khoury College of Computer Science, Northeastern University
shetty.sau@northeastern.edu, mittal.sid@northeastern.edu

Abstract

Continual learning remains a formidable challenge in machine learning, particularly in dynamic environments where models must adapt to new tasks without forsaking previously acquired knowledge. This study investigates diverse strategies to tackle this challenge, focusing on the SplitMNIST dataset. Three main approaches are explored: replay, regularization, and architectural adaptations. Greedy Sampler and Dumb Learner (GDUMB) leverage data sampling for replay, while Learning without Forgetting (LWF) employs distillation for regularization. The Continuous Weights with Re-Init (CWR) architecture facilitates continual growth of network capacity. Evaluation metrics include accuracy, stability, memory efficiency, and adaptability. Results highlight the strengths and limitations of each approach, offering insights into their complementary roles and potential synergies.

1 Introduction

Continual learning stands as a fundamental challenge in machine learning, particularly in scenarios where models must adapt to evolving data distributions over time (Kemker and Kanan 2017). This challenge is exemplified by the SplitMNIST dataset, a dynamic variant of the iconic MNIST dataset, where new classes are introduced sequentially while retaining previously learned ones (Lopez-Paz and Ranzato 2017). The ability to retain past knowledge while accommodating new information is critical for practical applications, motivating the exploration of various continual learning strategies.

We compare five continual learning approaches: naive (French 1999), cumulative (Kirkpatrick and et al. 2017), GDumb (Prabhu, Torr, and Dokania 2020), Learning without Forgetting (LwF) (Li and Hoiem 2018), and Copy-Weights with Re-init (CwR) (Lomonaco and Maltoni 2017). Each method addresses catastrophic forgetting and model adaptability differently.

The naive approach lacks adaptability (French 1999), while the cumulative approach continuously trains on old and new data, potentially facing scalability issues (Kirkpatrick and et al. 2017). GDumb penalizes parameter changes to mitigate forgetting (Prabhu, Torr, and Dokania 2020), while LwF transfers knowledge via distillation (Li and Hoiem 2018). CwR maintains representational capacity by constraining changes in learned representations

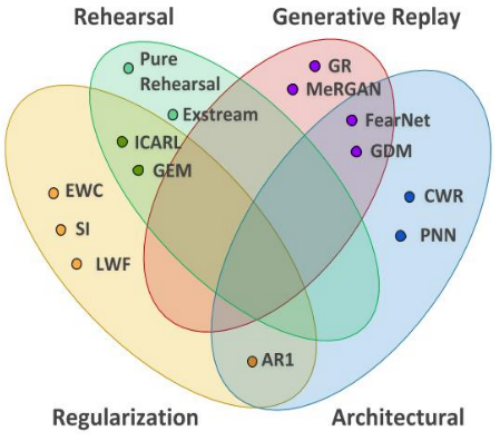

Figure 1: General Categorization of Continual Learning Algorithms by (Lange 2022)

(Lomonaco and Maltoni 2017).

Through extensive experimentation on SplitMNIST, we aim to understand each approach's performance characteristics. Our findings contribute to advancing continual learning methodologies, emphasizing the need for diverse strategies to build robust lifelong learning systems. By exploring replay, regularization, and architectural adaptations, we seek more effective solutions to real-world continual learning challenges.

2 Related works

Continual learning (CL) tackles the challenge of adapting machine learning models to evolving data distributions over time, a phenomenon known as concept drift. This is particularly pertinent in scenarios where models must incrementally adapt to new tasks without forgetting previously learned knowledge. Previous research in CL has explored various algorithms and strategies to mitigate catastrophic forgetting and adapt to evolving data distributions across different CL scenarios. These scenarios often include task incremental, class incremental, and domain incremental learning. In the realm of continual learning strategies, three main categories emerge - regularization, architectural, and re-

play/generative approaches. Regularization methods, such as Learning without Forgetting (LWF) (Li and Hoiem 2018), Elastic Weight Consolidation (EWC) (Kirkpatrick and et al. 2017), and Synaptic Intelligence (SI) (Zenke, Poole, and Ganguli 2017), constrain model parameters to preserve knowledge from previous tasks. Architectural strategies, including Multi-head architectures, (Kim, Ke, and Liu 2022), Copy Weights with Re-Init (CWR) (Lomonaco and Maltoni 2017), and Progressive Neural Networks (PNNs) (Rusu et al. 2016), allow models to dynamically adjust their capacity to accommodate new knowledge while retaining old representations. Replay and generative strategies, such as Greedy Sampler and Dumb Learner (GDUMB) (Prabhu, Torr, and Dokania 2020) and Maximally Interfered Retrieval (MIR) (Aljundi et al. 2019), leverage memory and generation capabilities to retrieve past examples and alleviate catastrophic forgetting. Hybrid approaches, such as Gradient Episodic Memory (GEM) [2] and Incremental Classifier and Representation Learning (iCARL) (Rebuffi et al. 2016), combine replay and regularization methods for enhanced performance. While some studies have investigated generic CL approaches, others have focused on domain-specific applications, particularly in image classification. Notable works (Chen and et al. 2018), (Mai et al. 2021) on online continual learning in image classification.

3 Background

In this section, we very briefly study some of the approaches used to build our models in this paper.

1. **Classical ML:** Our baseline model comprises a single-layer perceptron architecture (Figure 2) trained using Stochastic Gradient Descent (SGD) with cross-entropy as the loss metric. Each task consists of two classes from the MNIST dataset. During training, the model undergoes iterations on each task, and evaluation is conducted on all tasks to monitor both forward and backward accuracy. Early stopping is employed during training to prevent overfitting. It's important to note that this identical architecture is utilized across all subsequent approaches discussed below.

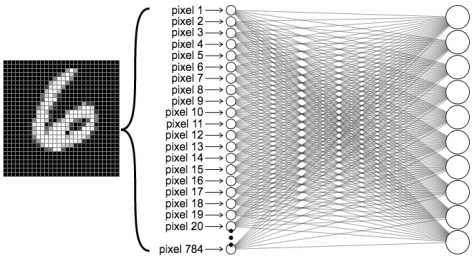

Figure 2: Classical ML approach using MLP

2. **Cumulative Learning:** In the cumulative learning framework, knowledge integration is structured around the tasks encountered sequentially. Rather than treating each task in isolation, the model incorporates new information by training a model on the data of all the tasks encountered till the current task, thereby enriching its understanding of the underlying concepts. This task-structured approach ensures that the model maintains a coherent representation of the entire learning trajectory, enabling seamless transfer of knowledge across tasks.

   The approach serves as a baseline for the continual learning models, comparing the approaches to the best possible accuracy metric with the selected model.

3. **Greedy Sampler and Dumb Learner (GDumb):** GDumb, is a straightforward yet effective approach devised to tackle the Continual Learning (CL) problem for classification tasks. At its core, GDumb adheres to a simple two-step process (Prabhu, Torr, and Dokania 2020):

   Greedy Sampling: During training, GDumb greedily stores samples from the incoming data stream into a memory buffer. The sampler ensures balanced representation of classes within the memory, thereby maintaining diversity and reducing bias. Notably, GDumb does not rely on predefined task boundaries or assumptions about the nature of incoming data.

   Dumb Learning: At inference time, GDumb employs a 'dumb' learner, i.e., a neural network trained from scratch using all the samples stored in the memory. This approach eschews complex adaptation mechanisms or task-specific optimizations, opting instead for simplicity and generality. In our experiment, we have chosen a buffer size of 500 and have used a multilayer perceptron as the learner model.

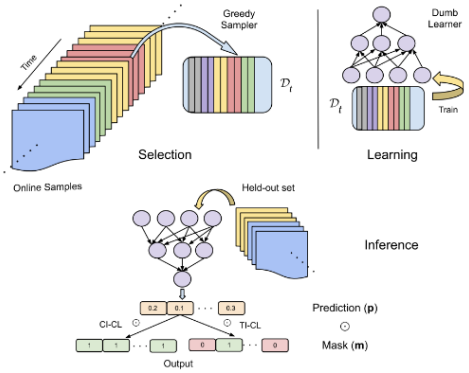

Figure 3: GDumb training methodology (Prabhu, Torr, and Dokania 2020)

4. **Learning without Forgetting (LWF):** Learning without Forgetting (LWF) utilizes a distillation-based regularization technique to prevent catastrophic forgetting while learning new tasks (Li and Hoiem 2018). It combines a distillation loss, which ensures the model's outputs for previously learned tasks remain stable, with the traditional cross-entropy loss for current tasks. This dual loss system encourages the model to maintain accuracy on older tasks by mimicking its previous outputs while adapting to new information. By preserving knowledge from past experiences and adapting to new challenges concurrently, LWF provides a balanced approach

Algorithm 1: Learning Without Forgetting

Given: θ_s (shared parameters), θ_o (task-specific parameters for each old task), X_n and Y_n (training data and ground truth for the new task)
Initialize:
  $Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // Compute output of old tasks for new data
  $\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // Randomly initialize new parameters
Train:
Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // Old task output
Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // New task output
$(\theta_s^*, \theta_o^*, \theta_n^*) \leftarrow \text{argmin}_{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}$
$\left[\lambda_o L_{\text{old}}(Y_o, \hat{Y}_o) + L_{\text{new}}(Y_n, \hat{Y}_n) + R(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n)\right]$

to continual learning, making it suitable for environments where tasks are presented sequentially and learning must be retained.

5. **Copy Weight with Re-init (CWR):** CWR strategy is designed for continuous learning from sequential batches, particularly under scenarios where tasks are not disjoint but constitute a single incremental task, such as class-incremental learning (Lomonaco and Maltoni 2017). The core idea behind CWR is to maintain two sets of weights for the output classification layer: consolidated weights (cw) used for inference and temporary weights (tw) used for training. During each training batch, the model is trained with stochastic gradient descent (SGD) on the classes present in that batch, with tw initialized randomly. After training, the weights in tw corresponding to the classes in the current batch are scaled and copied into cw. This process ensures that class-specific weights are learned without interference among batches.

4 Dataset

The SplitMNIST dataset is derived from the widely utilized MNIST dataset, which comprises grayscale images of handwritten digits ranging from zero to nine. However, unlike its predecessor, SplitMNIST introduces a dynamic setting wherein the original ten classes are sequentially partitioned into disjoint subsets, each constituting a distinct task. Specifically, SplitMNIST comprises a series of tasks, denoted as T_1,T_2,$T_3 \ldots T_n$ wherein each task is associated with a subset of the original MNIST classes. For instance, in a hypothetical scenario with n=5, the tasks may be delineated as follows: $T_1(0,1)$, $T_2(2,3)$, $T_3(4,5)$, $T_4(6,7)$ and $T_5(8,9)$. Notably, the partitioning of classes across tasks ensures progressive complexity, thereby simulating a realistic scenario of class incremental learning. The goal is to learn new tasks, while not deteriorating performance on the previous tasks, the older tasks are not provided again for training explicitly.

Algorithm 2: CWR (Copy Weight with Re-init)

Initialize:
$cw \leftarrow 0$ // initialize cumulative weights to zero
$past \leftarrow 0$ // track count of occurrences of each class
$\bar{\Theta} \leftarrow$ RANDINIT or LOAD_PRETRAINED() // initialize model parameters randomly or from a pre-trained model
For each training batch B_i:
  Expand output layer with neurons for new classes in B_i never seen before
  Define $tw[j] = \begin{cases} cw[j] & \text{if class } j \text{ is in } B_i \\ 0 & \text{otherwise} \end{cases}$
  Train the model with SGD on the classes of B_i:
    If $B_i = B_1$ learn both $\bar{\Theta}$ and tw
    Else learn tw while keeping $\bar{\Theta}$ fixed
  For each class j **in** B_i:
    Calculate $w_{\text{past}_j} = \frac{r_{\text{past}_j}}{c_{\text{cur}_j}}$, where c_{cur_j} is the number of patterns of class j in B_i
    Update $cw[j] = \frac{cw[j] \cdot w_{\text{past}_j} + (tw[j] - \text{avg}(tw))}{w_{\text{past}_j} + 1}$
    Update $past_j = past_j + c_{\text{cur}_j}$
Test the model by using $\bar{\Theta}$ and cw

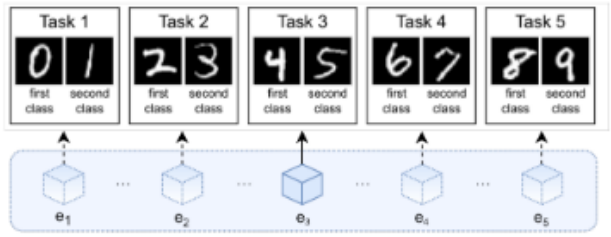

Figure 4: SplitMNIST dataset split across episodes

5 Experiment and Results

Experiment Setup

Our experimental setup involves training and evaluating the performance of our image classification model using the SplitMNIST dataset and the aforementioned continual learning algorithms. We measure key metrics such as in-domain, backward transfer, and forward transfer accuracies, RAM usage, Disk usage, Compute usage and Training speed per epoch of the models to assess the model's ability to adapt to evolving data distributions over time. The source code is provided here (https://github.com/KeenBean024/ContinualLearning) The models are trained on intel core i5-10300 CPU to track the compute carefully. They are implemented using Pytorch and avalanche library. All the approaches use the same Machine learning model and the architecture, an MLP model with a single hidden layer of 512 nodes. In all the approaches, the model is allowed to trained on the task, until early stopping if the loss doesn't decrease for 5 successive epochs. Backward Transfer Accuracy (BwT), Forward Transfer Accuracy (FwT), In-Domain Accuracy (IdA) are calculated in percent-

age between 0-100% with equations mentioned below, Free RAM is calculated as 8GB - (the memory in use), Free Disk and free CPU are imported from the Windows Operating system.

$$\text{BwT} = \frac{\sum_{i>j} R_{i,j}}{\frac{N(N-1)}{2}} \qquad \text{FwT} = \frac{\sum_{i<j} R_{i,j}}{\frac{N(N-1)}{2}} \qquad \text{IdA} = \frac{\sum_{i=1}^{N} R_{i,i}}{N} \tag{1}$$

Result

Model	FwT	BwT	IdA
Multilayer Perceptron	0.0	0.0	98.5
Cumulative model	0.0	96.4	96.8
MLP + GDUMB	0.0	94.19	92.5
MLP + LWF	0.0	12.9	99.0
MLP + CWR	0.0	85.9	90.5

Table 1: Accuracy comparison

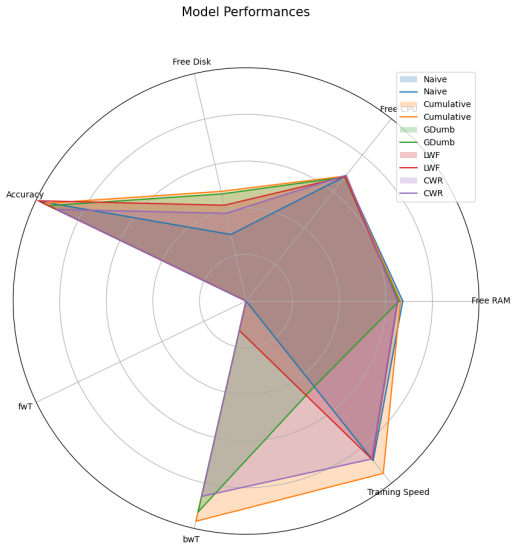

Figure 5: Model Performance comparison

Baseline model - Multilayer Perceptron

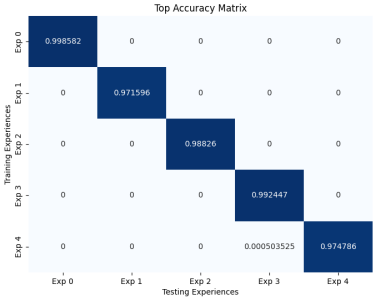

Figure 6: MLP Accuracy Matrix

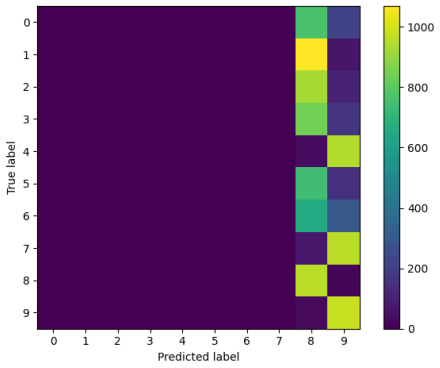

Figure 7: MLP Confusion matrix

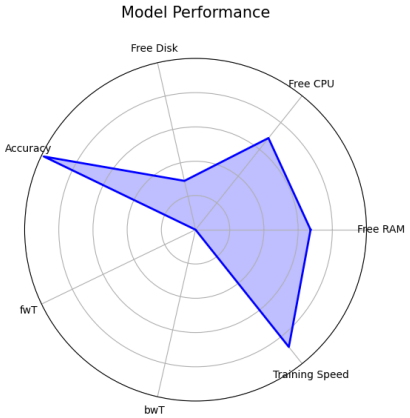

Figure 8: MLP Star diagram

The naive model exhibits near-perfect in-domain accuracy, indicative of high performance on the training data. However, both its forward and backward transfer accuracies are 0, suggesting a lack of generalization to future tasks and an inability to retain knowledge of past experiences, respectively. This phenomenon of catastrophic forgetting is further substantiated by the confusion matrix, which reveals exclusive prediction of the most recent class and neglect of earlier ones.

Cumulative model

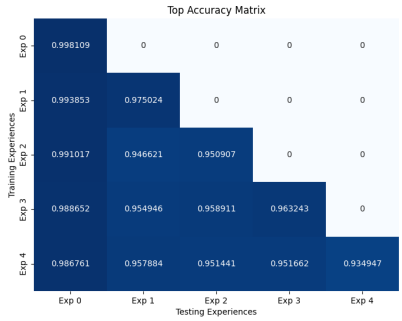

Figure 9: CM Accuracy Matrix

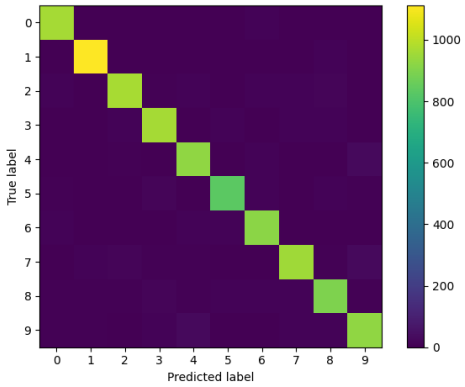

Figure 10: CM Confusion matrix

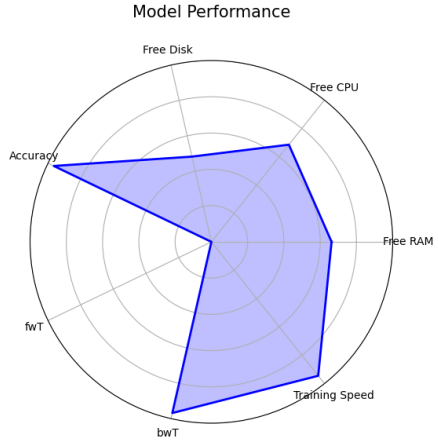

Figure 11: CM Star diagram

The above figure shows that the cumulative model achieves in-domain accuracy similar to the naive MLP but exhibits robust retention of past experiences, mitigating catastrophic forgetting, as confirmed by the confusion matrix analysis. Additionally, the star graph indicates that this model requires more training time and memory compared to the naive approach, a consequence of maintaining historical data for training.

Greedy Sampler and Dumb Learner (GDumb)

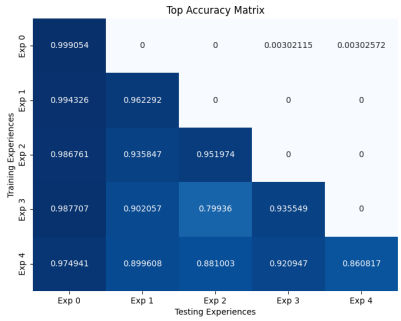

Figure 12: GDumb Accuracy Matrix

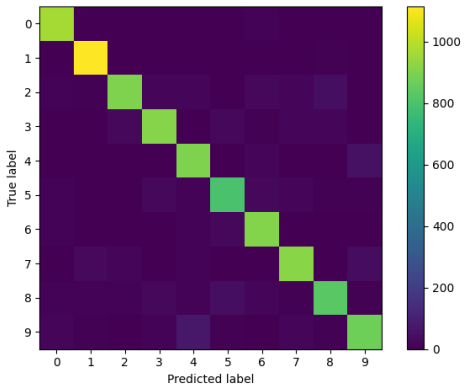

Figure 13: GDumb Confusion matrix

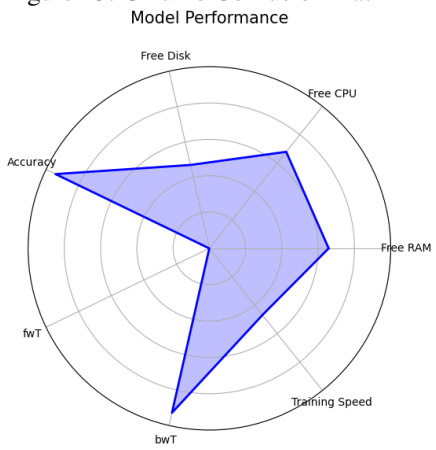

Figure 14: GDumb Star diagram

The GDumb approach seems to offer the best of both worlds. It combines the swift training speed of a naive MLP with the strong backward transfer accuracy seen in the Cumulative model. Plus, it achieves this with a smaller memory footprint, making it a smart compromise between the two approaches.

Learning without Forgetting (LWF)

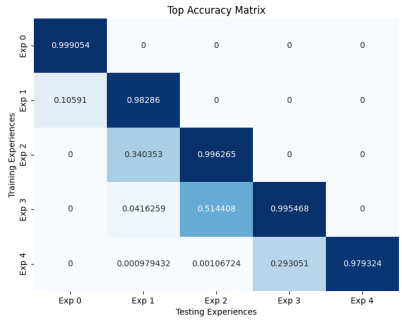

Figure 15: LWF Accuracy Matrix

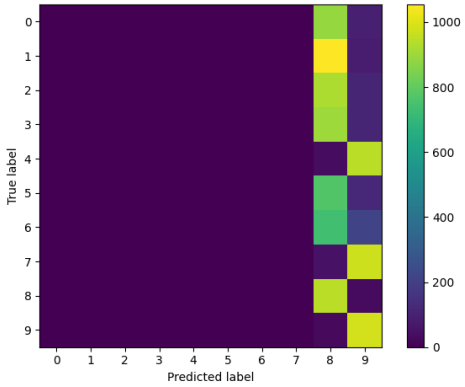

Figure 16: LWF Confusion matrix

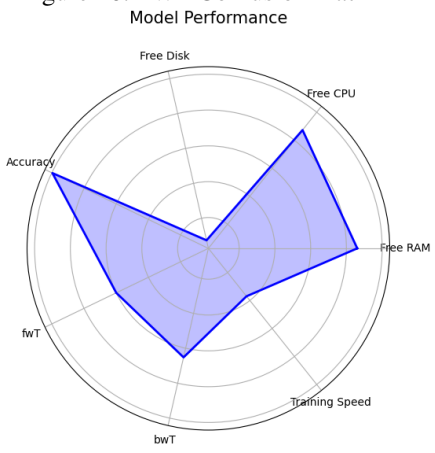

Figure 17: LWF Star diagram

Despite achieving high in-domain accuracy, LWF struggles to distill backward knowledge on the SplitMNIST dataset effectively. Indicating that knowledge distillation performs poorer than randomly sampling from history and training a fresh model on it. However, it is much faster to train and uses less memory than GDumb.

Copy Weight with Re-init (CWR)

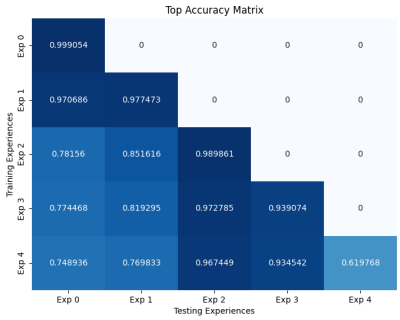

Figure 18: CWR Accuracy Matrix

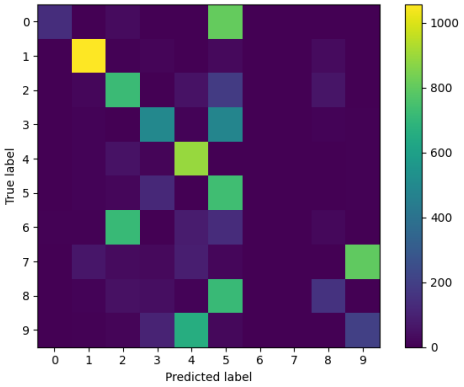

Figure 19: CWR Confusion matrix

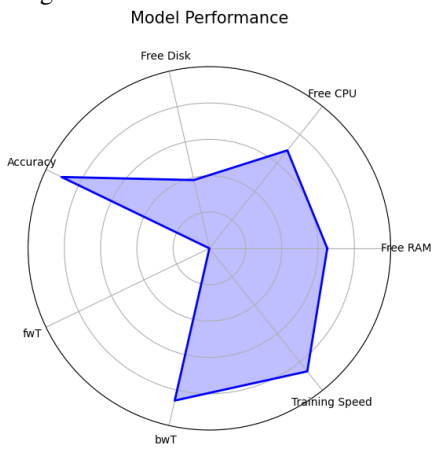

Figure 20: CWR Star diagram

Although CWR surpasses LWF, it lags behind GDUMB in achieving optimal backward accuracy. The architectural approach performs better than the knowledge distillation, however, it suffers due to disjoint task distribution and hyperparameter and other training methodologies.

6 Conclusion and Future Work:

This study rigorously evaluated four lifelong learning methodologies: the naive approach, cumulative learning, Learning without Forgetting (LWF), and CopyWeights with Re-init. Through meticulous experimentation, we observed that while the naive approach lacked robustness, cumulative learning and LWF effectively mitigated catastrophic forgetting and enabled seamless adaptation. While CopyWeights with Re-init struck a balance between knowledge retention and task-specific adaptation, it showed limitations in scenarios with substantial task variance. These findings emphasize the importance of principled lifelong learning methodologies for continual adaptation and knowledge retention in dynamic environments, paving the way for more robust continual learning artificial intelligence systems. In future, we will experiment on real-life scenario based datasets like CLEAR dataset (NOTE: We started with the dataset but lacked enough computational power to perform our experiments and debug code effectively) and hybrid approaches like AR1(Maltoni and Lomonaco 2018).

References

Aljundi, R.; Caccia, M.; Belilovsky, E.; Caccia, M.; Lin, M.; Charlin, L.; and Tuytelaars, T. 2019. Online Continual Learning with Maximally Interfered Retrieval. *arXiv preprint arXiv:1908.04742*.

Chen, Z.; and et al. 2018. *Lifelong Machine Learning*. Morgan & Claypool Publishers. Accessed 21 February 2024.

French, R. M. 1999. Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, 3(4): 128–135.

Kemker, R.; and Kanan, C. 2017. FearNet: Brain-Inspired Model for Incremental Learning. *arXiv preprint arXiv:1711.10563*.

Kim, G.; Ke, Z.; and Liu, B. 2022. A Multi-Head Model for Continual Learning via Out-of-Distribution Replay. *arXiv preprint arXiv:2208.09734*.

Kirkpatrick, J.; and et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

Lange, M. D. 2022. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 7, pp. 3366-3385*.

Li, Z.; and Hoiem, D. 2018. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 2935–2947.

Lomonaco, V.; and Maltoni, D. 2017. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, 17–26.

Lopez-Paz, A.; and Ranzato, M. 2017. Gradient Episodic Memory for Continual Learning. *arXiv preprint arXiv:1706.08840*.

Mai, Z.; Li, R.; Jeong, J.; Quispe, D.; Kim, H.; and Sanner, S. 2021. Online Continual Learning in Image Classification: An Empirical Survey. *arXiv preprint arXiv:2101.10423*.

Maltoni, D.; and Lomonaco, V. 2018. Continuous Learning in Single-Incremental-Task Scenarios. *arXiv preprint arXiv:1806.08568*.

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 511–527.

Rebuffi, S.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2016. ICaRL: Incremental Classifier and Representation Learning. *arXiv preprint arXiv:1611.07725*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *arXiv preprint arXiv:1606.04671*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. *arXiv preprint arXiv:1703.04200*.